Occluded Object Search by Relational Affordances

Bogdan Moldovan[1] and Luc De Raedt[1]

***Abstract*— Searching for objects in occluded spaces is one of the problems robots need to solve when tackling mobile manipulation tasks. Most approaches focus only on searching for a specific object. In this paper, we use the concept of relational affordances to improve occluded object search performance. Affordances define action possibilities on an object in the environment and play a role in basic cognitive capabilities. Relational affordances extend this concept by modelling relations between multiple objects. By learning and using a relational affordance model we can search for any of the multiple objects that afford a given action, each object type having a probability distribution over possible sizes and shapes, and where spatial relations between objects such as co-occurrence and stacking are modelled. The experimental results show the viability of the relational affordance models for occluded object search.**

I. Introduction

The goal of robotics is to develop mobile, physical agents capable of reasoning, learning and manipulating their environment. To achieve a task, a robot first needs to find certain objects in the environment to manipulate. However, in a real life indoor environment, most objects are not immediately visible, but lie on shelves or cupboards behind other objects. If the searched object is not detected, the robot needs to reason about which objects to remove to continue its search.

Previous research in searching for occluded objects [7], [25] focused on object search in environments where all objects have known shapes and sizes, and lie on a flat surface. This is a practical approach in a known environment where only using a particular object can achieve a task. Imagine being in an unknown kitchen and wanting to drink water. We look at shelves and search for any object to use: it can be either a mug, or a glass, or maybe a plastic cup. Moreover, we do not know the exact shapes and sizes of these objects, although we posses prior knowledge of them. Finally, in this environment not all objects are on a flat surface: plates are stacked on top of each other, and the mug is on top of them.

A promising approach for the development of robots' skills is learning *object affordances*, which capture *action opportunities* to structure the environment, and model relations between three variables: objects properties, actions, and effects [12], [15]. They follow the robotics developmental framework, which proposes acquiring new skills on top of old ones by experimentation in the environment [13].

Recent work [14] extended affordance models to relational affordances by modelling interactions and relations between objects with the help of statistical relational learning (SRL) [5], [4]. SRL combines logical representations, probabilistic reasoning and machine learning, and was used to generalize and enable inference in scenarios modelling spatial relations between multiple objects [14]. In this paper, we use relational affordances to search for an object that affords a given action.

A. Problem Statement and Approach

The robot is in a kitchen environment similar to [25], with the space partitioned in a finite set of disjoint containers with contents independent from each other, which we name shelves. Each shelf contains a set of objects, only some of which are visible to the robot, while the rest are occluded. The robot can remove the visible objects from a shelf, causing a subset of the previously occluded objects to become visible. Each object is associated with a set of actions it affords. Fig. 1 shows an example of such an environment. The robot's task is to find an object affording a given action a by removing as few objects as possible.

Fig. 1: Kitchen shelves with visible and occluded objects.

We make the following additional assumptions, similar to [25]: the environment is static, apart from any robot manipulation (i.e., object removal). Each shelf has known geometry. Each visible object is recognizable, so given a sufficiently clear view its type can be resolved without error. In addition to [25], we allow for objects to lie on other objects in a shelf (e.g., a stack of plates).

Let O_a represent the set of all objects affording action a. The robot proceeds as follows: it observes the set of shelves S. In each shelf $s_i \in S$ it notes the set of visible objects O_{vis_i}. For each s_i it computes, with the help of a relational affordance model, the probability that objects o in that shelf afford a: $P(o\ in\ s_i, o \in O_a | O_{vis_i})$. The robot chooses the shelf with the highest probability and removes its visible objects, causing some occluded objects to become visible. The process is repeated until an object is found.

Compared to [25], relational affordance models allow a search setting where object shape and size is not fixed, but modelled by probability distributions, where more spatial relations can be easily modelled (e.g., stacking), and objects are associated to (and searched for) the actions they afford.

To build the relational affordance model, the robot is presented with: i) a set of objects, given by their properties

[1]Bogdan Moldovan and Luc De Raedt are with the Department of Computer Science, Katholieke Universiteit Leuven, Belgium

Bogdan Moldovan is supported by IWT (agentschap voor Innovatie door Wetenschap en Technologie). This work is supported by the European Community's 7th Framework Programme, grant agreement First-MM-248258.

(length, width, height, bounding shape and type label), and the list of afforded actions; ii) a set of images of shelves, where each object is labelled with its type, and from which a set of probabilities of object co-occurrences and stacking are extracted, and iii) background knowledge (i.e., logical rules) about object spatial relations and afforded actions. Once this model is learnt, the occluded space is modelled, and then we can perform inference to compute the required probabilities.

B. Contributions and Outline

Previous work on searching for occluded objects [25], [7] focused on finding a specific given hidden object. The work of [25] uses object co-occurrence information and spatial constraints to build a model used to search for a specific object in a space populated by objects from a finite set of known types with fixed known shape and size. Our main contribution is using the concept of relational affordances [14] to search for any object affording a given action. Multiple object types can afford the action, and each type allows for many different objects with size and shape modelled by probability distributions, thus relaxing some of the assumptions of previous work. Moreover, we allow for stacked objects, a more realistic modelling of objects in shelves, introducing more complex object spatial relations.

The use of SRL methods in relational affordances allows to easily model the object properties probability distributions and the relations to the actions they afford, as well as the spatial relations between objects (e.g., stacking) than individually modelling each constraint. Additionally, the model can be used as-is for different inference tasks without any need for additional modelling (e.g., computing the probability of two objects affording two different actions being on the same shelf, useful when moving between shelves is costly). Using simple affordance models using Bayesian Networks (BNs) as in [12], [15] is infeasible in a setting with so many objects.

This paper is organized as follows: Section II presents related work, Section III affordance-based models, and Section IV the main contribution of this paper: modelling and using relational affordances for occluded object search. Section V presents experimental results and we conclude in Section VI.

II. Related Work

Affordances were introduced by J. J. Gibson [8] and are used to model world-robot interaction. They are used in several settings, e.g., in the context of imitation learning [12], [15]. Relational affordance models for multiple object settings were introduced in [14], based on probabilistic relational models and SRL [4], [6].

Work on object search tackled manipulating sensors for better visibility of the searched object [26], [22], [23]. Object co-occurrence and object-scene context information for improving object search were explored in settings as [11], [19], [21]. Using background knowledge from previously seen, similar environments, to improve object search in unknown environments was studied in [10]. The use of spatial relations was studied for creating search strategies in multiple-room environments [2]. Exploiting domain knowledge for object discovery was considered in [3]. Learning hidden affordances based on object geometry and pose was studied in [1]. Lastly, systems such as KnowRob [24] integrate knowledge representation and reasoning into general robotic control.

Recent work on occluded object search by manipulation includes a search algorithm for optimally finding a hidden object in a table-top setting [7] and search for a hidden object in a setting with multiple containers with both visible and hidden objects [25], the latter being the setting we extend.

III. Affordance-based Models

Affordance models [8] model the robot-world interaction by capturing *action opportunities* to structure the environment, e.g., a glass is handled in a different way than a book. An affordance is an intrinsic property of an object, allowing an action to be performed with the object by the agent performing the action [18]. For example, in Fig. 2 a newspaper, a book and a tablet afford reading, but out of these only a tablet affords playing a game.

Fig. 2: Actions afforded by several objects

Affordances define relationships between the robot and environment through the robot's sensing and motor capabilities [12], [15]. When the robot explores its environment, it can build an affordance model to represent the correlations between the set of objects properties detected by its sensors, the repertoire of actions available to it, and the effects of performing the actions. However, perceiving an affordance does not mean the agent has to act upon it, acting being required only when an effect needs to be achieved [18].

A. Single Object Affordances

Affordances are generally learned by the robot by exploring the environment and manipulating objects [12], [15]. In our setting, we are interested in modelling the relations between object properties and the afforded actions. For showcasing our approach, we use the following object properties relevant to our search setting: length, width, height, shape (prism or cylinder) and type. Others can also be considered.

For learning the affordance model we use a set of IKEA kitchen objects taken from the on-line catalog[1]. We picked six object types from the corresponding IKEA categories: *glasses*, *cups*, *pitchers*, *plates*, *bowls* and *serving dishes*. We used 10 objects of each type, with the exception of pitchers from which only 5 were used. Fig. 3 shows all the 55 objects.

In our kitchen setting, we consider the six afforded actions: *water drinking*, *coffee drinking*, *pouring*, *dinner serving*, *soup serving* and *appetizer serving* (more can be added). Actions are afforded by objects from multiple types, e.g., water drinking is afforded by glasses as well as big cups. Table I outlines all actions and the objects that afford them.

[1] http://www.ikea.com/us/en/catalog/categories/departments/eating/

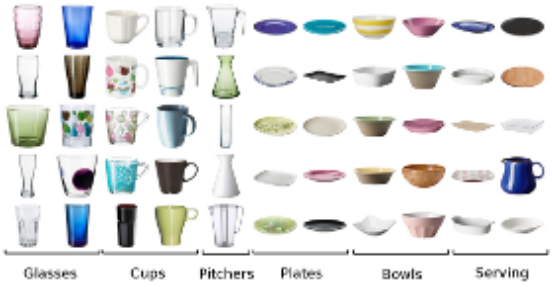


Fig. 3: IKEA objects used for learning affordance model (Note: object images are not to same scale)

TABLE I: Actions and the objects that afford them.

Action	Objects
Water drinking	Glasses Big cups (volume $> 25cl$)
Coffee drinking	Cups
Pouring	Pitchers Big glasses (volume $> 40cl$) Gravy boat serving dish (height $> 10cm$)
Dinner Serving	Plates Shallow bowls (height $< 5cm$) Small serving dishes (length $< 30cm$)
Soup Serving	Bowls
Appetizer Serving	Serving dishes (w/o gravy boat) Big plates (length $> 30cm$) Big, shallow bowls (length$> 30cm$,height$< 5cm$)

In a real-world setting the robot is presented with a set of objects to observe for which it extracts the above mentioned properties. A system such as BLORT [16] can be used to model the objects by simple shapes. For each object, a human can tell the robot the afforded actions, taking into consideration the robot's available motor capabilities. This list of afforded actions can be adjusted by exploratory manipulation of the objects by the robot. For ease of exposition, we assume the robot is presented directly with the set of object properties and list of afforded actions for the 55 objects.

For representing the probability distributions of the object properties we learn and use a BN. In our setting, the conditional dependencies between the object properties are shown in Fig. 4. For the length and height we learn normal distributions for each object type from our 55 objects training data. For example, we obtained for the length of a glass $\mathcal{N}(6.05, 0.4742)$, for the height of a pitcher $\mathcal{N}(22.8, 4.6583)$, etc. (all numbers are cm). For the shape, we learn, depending on each type, the probability of being a prism or a cylinder, e.g., the shape of a plate has a probability of 83.33% of being a cylinder, and 16.67% a prism. The width is equal to the length if the shape is a cylinder, otherwise it is approximated with the normal distribution $\mathcal{N}(22.6364, 6.5463)$, whose parameters were similarly learnt from the training data. For modelling the actions, we use the logical rules from Table I.

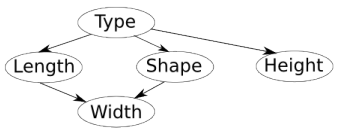


Fig. 4: Conditional dependencies between object properties

We now have a single object affordance model for our setting. To extend it to a relational domain to be used for searching for occluded objects, we need first to model it with a *probabilistic programming language* (PPL) as in [14].

IV. Relational Models for Affordances

In this section we describe our main contribution: modelling and using relational affordances for occluded object search. Previously [14] affordance models were extended with relations such as relative distance or angle of orientation between objects in order to perform action prediction in a multiple object environment. For searching for occluded objects, we are interested instead in extending the affordance model with the co-occurrence relation, as well as spatial relations (i.e., "on") between objects. Thus, as a difference to [14], we do not learn and then generalise a second BN for a multiple object setting, but instead we model the co-occurrence and stacking probabilities and relations, and use logical rules to restrict the spatial configurations of objects.

A. Probabilistic Programming Languages

A PPL is a programming language specifically designed to efficiently describe and reason with probabilistic relational models. To do this in a PPL, one writes a program which consists of a set of probabilistic facts and a set of logical rules (which express *domain knowledge* and *constraints*).

Previously [14] we modelled relational affordances using the PPL ProbLog [6]. However, using relational affordances for object search is different since we deal with continuous distribution random variables (e.g, length, etc.), modelled by normal distributions. Distributional Clauses (DCs) [9] are a better candidate for our statistical relational representation.

We first review basic concepts of logic programming: An atom $pred(t_1, ..., t_n)$ consists of a predicate $pred/n$ of arity n and t_i terms. A term is either a (lowercase) *constant*, (uppercase) *variable*, or functor $func/n$ applied on n terms. A *ground* atom does not contain any free variables. A *definite clause* is an expression of the form $h \leftarrow b_1, ..., b_n$, where h and b_i are atoms. It states that h is true whenever all b_i are true. If $n = 0$ we have a fact $f \leftarrow$, i.e., f is true. A *substitution* $\theta = \{X_1 = t_1, ..., X_n = t_n\}$ maps each variable X_i to a term t_i. Applying a substitution θ to an atom a yields $a\theta$, where each occurrence of X_i in a is replaced with t_i.

DCs [9] are an extension of the distribution semantics of [20]. DCs allow one to define random variables with any distribution, continuous or discrete. A DC is an expression of the form $h \sim \mathcal{D} \leftarrow b_1, ..., b_n$, where b_i are atoms and $\sim$ a binary predicate written in infix notation. In a DC, each ground instance of the clause $(h \sim \mathcal{D} \leftarrow b_1, ..., b_n)\theta$, for a substitution θ, defines the random variable $h\theta$ being distributed according to distribution $\mathcal{D}\theta$ when all $b_i\theta$ hold.

To model the length of a glass with probability distribution $\mathcal{N}(6.05, 0.4742)$, from the previous section, one can write:
$\texttt{length}(\texttt{O}) \sim \texttt{gaussian}(6.05, 0.4742) \leftarrow \texttt{type}(\texttt{O}, \texttt{glass})$.
with variable `O` universally quantified over the set of all objects; atom $\texttt{type}(\texttt{O}, \texttt{glass})$ true if the type of `O` is a glass.

The probability distribution of the shape of a plate:
$\texttt{shape}(\texttt{O}) \sim \texttt{finite}([0.8333 : \texttt{cyl}, 0.1667 : \texttt{prism}]) \leftarrow \texttt{type}(\texttt{O}, \texttt{plate})$.

The term $\mathcal{D}$ that represents the distribution can be non-ground: values, probabilities, or distribution parameters can be related to conditions in the body. Additionally, the term $\simeq(d)$ represents the value of the random variable d.

So, we can encode the fact that the width of a cylinder object is the same as the length as follows:

$\texttt{width(O)} \sim \texttt{finite}([\texttt{1.0} : \texttt{L}]) \leftarrow \simeq(\texttt{shape(O)}) == \texttt{cyl},$
$\texttt{L is} \simeq(\texttt{length(O)}).$

We define the actions the objects afford with the help of definite clauses. We illustrate this for the *pouring* action:

$\texttt{action(O, pour)} \leftarrow \texttt{type(O, pitcher)}.$
$\texttt{action(O, pour)} \leftarrow \texttt{type(O, glass)}, \texttt{objVol(O, V)}, \texttt{V} > 400.$
$\texttt{action(O, pour)} \leftarrow \texttt{type(O, serving)}, \simeq(\texttt{height(O)}) > 10.$

where we previously defined atom $\texttt{objVol(O, V)}$ to unify variable V to the volume in cm^3 of object O.

The rest of the affordance model is modelled using DCs in a similar manner. Once the program is defined, the inference algorithm based on sampling from [9] or [17] is used to compute the probability of a user's query.

B. Co-occurrence and Stacking Spatial Relations

Object type co-occurrence in a scene was used to improve object search results [10], [2] and to create a generative model for searching for occluded objects [25]. We use it as one of the relations to extend our affordance model.

There are several ways for a robot to learn co-occurrence probabilities of different object types. In an affordance setting, or a developmental framework, which propose acquiring skills by experimentation and interaction with the environment, it can extract these probabilities by observing many environments with shelves. Alternatively, such probabilities were obtained from the Web [19], using semantic information obtained from search results. In our approach, we use labelled shelf images, which allows learning from realistic kitchen object distributions similar to a developmental framework approach. We used 66 shelves from 20 kitchen shelves images from Google Images. We labelled each object with its type, ignoring objects whose type was not one of the six modelled ones. Fig. 5 shows some of the shelves.

Fig. 5: Sample shelf images from which co-occurrence and stacking probabilities were learnt

Since we want to find the shelf with the highest probability of having objects with certain properties, we choose to model the concept of object co-occurrence by the probability of an object on a shelf being of a certain type given that on that shelf there are objects of another given type. We obtain probability values from the labelled shelf images by maximum likelihood estimation. For example, when observing a bowl:

$\texttt{type\_ifobs(O, bowl)} \simeq \texttt{finite}([\texttt{0.0798} : \texttt{glass},$
$\texttt{0.1007} : \texttt{cup}, \texttt{0.0703} : \texttt{pitcher}, \texttt{0.3583} : \texttt{plate},$
$\texttt{0.3672} : \texttt{bowl}, \texttt{0.0237} : \texttt{serving}]) \leftarrow \texttt{true}.$

which means that in a shelf with bowl(s) there is a probability of 7.98% of an object O being a glass, and so on.

If the observed object types (later given as evidence) on the shelf are encoded by the random variables $observed\_type(O)$, the type of the unknown occluded object O is given by the following logical rule:

$\texttt{type(O, T)} \leftarrow \simeq(\texttt{observed\_type(O2)}) == \texttt{T2},$
$\simeq(\texttt{type\_ifobs(O, T2)}) == \texttt{T}.$

Object O has a probability $\texttt{type\_ifobs(O, T2)}$ of being of type T whenever we observe an object of type T2.

Similarly, stacking probabilities are learnt and used. From the same shelf images, by maximum likelihood estimation, we learn the probability of an object being of a certain type given that it is on top of another object of a given type. For example, the probability for the object type on top of a plate:

$\texttt{ontype(O, plate)} \simeq \texttt{finite}([\texttt{0} : \texttt{glass}, \texttt{0.0024} : \texttt{cup},$
$\texttt{0} : \texttt{pitcher}, \texttt{0.8676} : \texttt{plate}, \texttt{0.0284} : \texttt{bowl},$
$\texttt{0} : \texttt{serving}, \texttt{0.1016} : \texttt{none}]) \leftarrow \texttt{true}.$

where *none* signifies the probability that there is nothing on top of a plate. Then these stacking probabilities can be used in a logical rule similar to the one for co-occurrence.

C. Overall Model and Spatial Constraints

To use our defined model for inferring the probability that an object with the required affordance action is found in the occluded space we need to also model the occluded space.

We model the occluded space, with shape and size assumed known, with the help of logical rules. The packed area of an object is given by the $packedArea/2$ predicate. The total area is defined recursively, assuming a greedy object packing strategy. Different area modellings are also possible. Here is a simplified version of a model:

$\texttt{area(AS, [])} \leftarrow \texttt{AS} <= \texttt{0}.$
$\texttt{area(AS, ObjList)} \leftarrow \texttt{objShorterThan(O, MaxHeight)},$
$\texttt{RemainH is MaxHeight} - \simeq(\texttt{height(O)}),$
$\texttt{stackOnTop(O, RemainH, StackObjList)},$
$\texttt{packedArea(O, AObj)}, \texttt{NewA} = \texttt{AS} - \texttt{AObj},$
$\texttt{area(NewA, NewObjList)},$
$\texttt{append([O|StackObjList], NewObjList, ObjList)}.$

For modelling height constraints, we assumed the robot vision sensor (e.g., camera) is at the same height as the shelf, so the maximum height of a stack of occluded objects is the maximum observed stack height in the shelf. Models where the camera looks at the shelf under an angle are also possible.

Spatial constraints can be modelled with logical rules. For example, bigger objects cannot be put on top of smaller ones:

$\texttt{checkStackConstraints(OBottom, OTop)} \leftarrow$
$\simeq(\texttt{length(OTop)}) <= \simeq(\texttt{length(OBottom)}),$
$\simeq(\texttt{width(OTop)}) <= \simeq(\texttt{width(OBottom)}).$

where the $checkStackConstraints$ atom will be used in the body of the clause defining $stackOnTop$.

As stated before, the probability of objects o affording action a in a shelf s_i is given by $P(o\ in\ s_i, o \in O_a | O_{vis_i})$, O_a the set of all objects affording the action, and O_{vis_i}

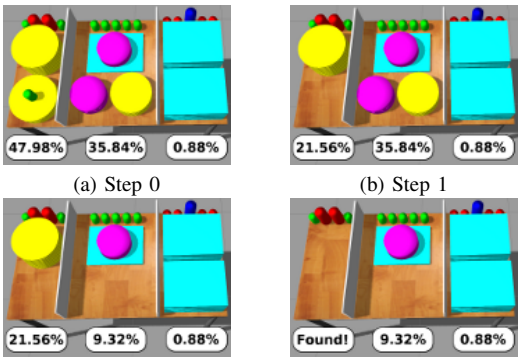
the set of visible objects in s_i. To compute this probability using our relational affordance model, assuming an occluded space of size as in s_i, we use DCs with the inference methods of [9], [17] to compute the probability of the query: $\texttt{area}(\texttt{as}, \texttt{ObjList}), \texttt{member}(\texttt{Obj}, \texttt{ObjList}), \texttt{action}(\texttt{Obj}, \texttt{a})$. given the evidence formed by a conjunction of $observed\_type$ atoms. To compute which shelf is more likely to contain an occluded object which affords the given action, we compute $\arg\max_{s_i} P(o \; in \; s_i, o \in O_a | O_{vis_i})$. This shelf is explored next by removing its visible objects.

Our relational model offers added flexibility since using the same model we can ask for the probability of two objects on the same shelf affording two different given actions, a_1, a_2, by asking the query: $\texttt{area}(\texttt{as}, \texttt{ObjList})$, $\texttt{member}(\texttt{O}, \texttt{ObjList}), (\texttt{action}(\texttt{O}, \texttt{a}_1) \; ; \; \texttt{action}(\texttt{O}, \texttt{a}_2))$.

V. Evaluation and Results

We test our relational affordance model for occluded object search. We first show a small setting in simulation, then we investigate in larger settings how it compares with searching for a specific object (considering co-occurrence and spatial relations), and with baseline systematic searches.

We integrated our model with the Gazebo robot simulator. We set up an environment with three shelves with objects, as in Fig. 6a. Objects are modelled by our two shapes. Object colours denote object type: glasses are red, cups green, pitchers blue, plates yellow, bowls magenta, serving dishes cyan. Only the front layer of objects (bottom of image) is visible to the robot, the rest is occluded. Similar to [25], our assumption is that a visible object type can be resolved without error, so we abstract the sensing by a program returning visible layer object types of a queried shelf.



(a) Step 0 (b) Step 1

(c) Step 2 (d) Step 3

Fig. 6: Simulation of search for an object affording pouring.

Fig. 6 and the supplementary video show the step-by-step process of finding an object which affords pouring. This can be either one of the two big glasses in the back of the left shelf, or the pitcher in the back of the right shelf. The probabilities capture the likelihood that there is an object on that shelf affording pouring (since there can be more than one such object in all the shelves, or even none, these probabilities do not add up to one). Serving dishes very rarely co-occur with objects that afford pouring, and just a few (i.e., gravy boat) afford pouring themselves.

From initial setting (Fig. 6a), the search proceeds as follows:

1) Fig. 6a: the left shelf is the most likely to contain an object affording pouring. Its front layer is removed. No new object types are observed, while the available space decreases, causing the probability to drop.
2) Fig. 6b: the centre layer has highest probability, its front layer is removed. Serving dishes are observed, which would barely increase probability, but the remaining available space, especially observed height, is greatly reduced, causing a big drop in the probability.
3) Fig. 6c: the left shelf is again the most likely one to contain such object, and its next layer is removed.
4) Fig. 6d: object affording pouring is found in left layer.

We investigate more thoroughly the benefits of using our model. The setting for this experiment is based on the one in [25]. Our environment was setup as follows: we picked 10 shelf images from Google Images (similar to Fig. 5, but with different object type distributions) not used for learning the model parameters, and labelled the object types in each. The shelves contain anywhere from 3 to 27 objects. We setup each shelf to have three different layers, the first visible, and the other two occluded behind it (similar to Fig. 6a).

From each of these shelf images, we obtain 10 different shelf configurations by randomly sampling each object into one of the three layers, while keeping the same stacking configuration. Then, for each object we sample a shape and size for it from our learned distributions, with the exception of stacked identical objects in the image, where we use the shape and size of the bottom object for the ones on top. So we generated a total of 100 shelves.

Our test setting consists of environments of 10 shelves, randomly picked each time from the 100 generated shelves. We pick an action to search for that is afforded by at least one occluded object, but by none of the initial visible ones (otherwise the search will terminate immediately). We generate and run 1000 different such object search scenarios.

We compare using the relational affordance model to a search for a specific object taking into consideration the same spatial relations (equivalent to modelling [25] together with stacking and height constraints), and with random search baselines for affordances and exact objects. These baselines, referred to as systematic to keep with the name convention of [25], pick a shelf at random and empty it until the target is found. In each scenario, the system uses one of its search strategies to pick the shelf to search next. Since we do not know where in an occluded layer the target might be, we remove the whole front layer to reveal the occluded objects behind it. The process continues until the target is found.

To summarize, the compared four search strategies are:

- **Relational affordance model**
- **Relational exact search**: use the model with spatial relations to look for shelf most likely to have object with given type and shape, then in the layer search for specific object size (querying for size directly not possible since it is represented by continuous distributions)
- **Systematic affordance search**: shelves are chosen at random and emptied layer by layer until an object affording the given action is found

- **Systematic exact search**: shelves are chosen at random and emptied layer by layer until exact object found

The results are shown in Fig. 7. The figure shows a cumulative plot of the number of objects moved before the target is found. The more to the left a line is, the better.

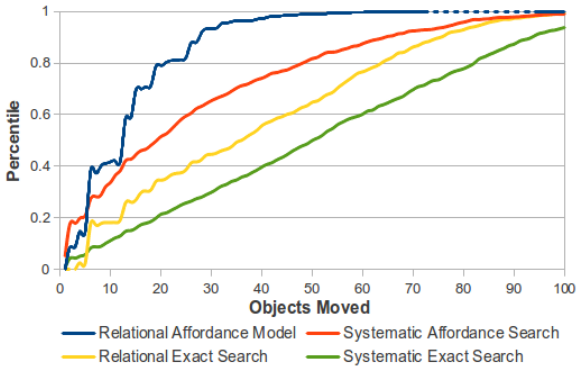


Fig. 7: Comparison of the four search strategies: percentile of simulation trials finding an object within a number of moves

For example, when using the relational affordance model, the median number of objects that need to be moved is 13, while 95% of the trials moved 32 objects or less. Using our relational affordance model is the best approach for minimising the number of objects moved. The second best approach is the systematic search for an affordance. Then comes the search for an exact object taking into account spatial object relations, and finally the systematic exact search. We can also deduce that incorporating object affordances into object search models is the most beneficial aspect, as the systematic search for an affordance performs better than a search for the exact object taking into account spatial relations.

Experiments were run on computers with Intel Core $i5-2500$ $3.3GHz$ processors, $6MB$ cache, and $8GB$ memory. We implemented our model with DCs and used 500 samples for computing query probabilities. Computing $P(o\,in\,s_i, o \in O_a|O_{vis_i})$ for one shelf s_i took on average 3.3 seconds.

Fig. 8(l) shows how run-time for obtaining one sample varies with the number of object types. We increased the number of types and actions by groups of six. If each action is afforded by only few objects, we need to increase the number of samples with the number of actions for the same relative precision, as each affordance now becomes more unlikely. Fig. 8(r) shows how run-time varies with the occluded area size for the 6 object types setting. Occluded shelf size for the experiments in Fig. 7 was around $0.2m^2$.

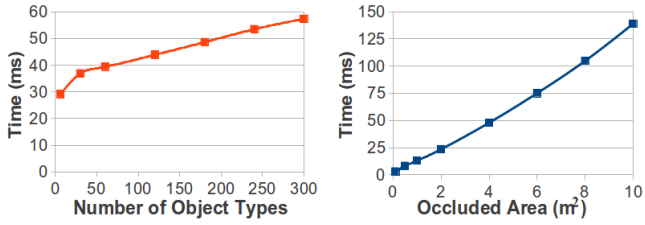


Fig. 8: Run-time in milliseconds to obtain one sample while varying: (l) number of object types, (r) occluded search area

VI. Conclusion and Future Work

We presented an approach for learning and using a relational affordance model for occluded object search, and showed that it can be used successfully for improving search performance compared to searching for a specific object. As future work, we will model other spatial relations (e.g., "near": a salt shaker is likely near a pepper shaker), or organisational principles as in [21]. Domain knowledge can be added for modelling other environments (e.g., living room). Uncertainty in the visible object types, or partial visibility constraints, can also easily be modelled.

References

[1] A. Aldoma, F. Tombari, and M. Vincze. Supervised learning of hidden and non-hidden 0-order affordances and detection in real scenes. In *ICRA*, pages 1732–1739. IEEE, 2012.

[2] A. Aydemir, K. Sjöö, J. Folkesson, A. Pronobis, and P. Jensfelt. Search in the real world: Active visual object search based on spatial relations. In *ICRA*, pages 2818–2824. IEEE, 2011.

[3] A. Collet, B. Xiong, C. Gurau, M. Hebert, and S. Srinivasa. Exploiting domain knowledge for object discovery. In *ICRA*. IEEE, 2013.

[4] L. De Raedt. *Logical and Relational Learning*. Springer, 2008.

[5] L. De Raedt and K. Kersting. Probabilistic inductive logic programming. In *Prob. Ind. Log. Progr.*, pages 1–27, 2008.

[6] L. De Raedt, A. Kimmig, and H. Toivonen. Problog: A probabilistic Prolog and its application in link discovery. In *IJCAI*, 2007.

[7] M. Dogar, M. Koval, A. Tallavajhula, and S. Srinivasa. Object search by manipulation. In *ICRA*. IEEE, 2013.

[8] J. J. Gibson. *The Ecological Approach to visual perception*. Boston: Houghton Mifflin, 1979.

[9] B. Gutmann, I. Thon, A. Kimmig, M. Bruynooghe, and L. De Raedt. The magic of logical inference in probabilistic programming. *CoRR*, abs/1107.5152, 2011.

[10] D. Joho, M. Senk, and W. Burgard. Learning search heuristics for finding objects in structured environments. *RAS*, 59(5):319–328, 2011.

[11] T. Kollar and N. Roy. Utilizing object-object and object-scene context when planning to find things. In *ICRA*, pages 2168–2173. IEEE, 2009.

[12] M. Lopes, F. S. Melo, and L. Montesano. Affordance-based imitation learning in robots. In *IROS*, pages 1015–1021. IEEE, 2007.

[13] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini. Developmental robotics: A survey. *Connection Science*, 15:151–190, 2003.

[14] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt. Learning relational affordance models for robots in multi-object manipulation tasks. In *ICRA*. IEEE, 2012.

[15] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory-motor coordination to imitation. *IEEE Transactions on Robotics*, 24:15–26, 2008.

[16] T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze. BLORT - The Blocks World Robotic Vision Toolbox. In *Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.

[17] D. Nitti, T. De Laet, and L. De Raedt. A particle filter for hybrid relational domains. In *IROS*. IEEE, 2013.

[18] E. Rome, P. Doherty, G. Dorffner, and J. Hertzberg. Towards affordance-based robot control. In *Towards Affordance-Based Robot Control*, number 06231 in Dagstuhl Seminar Proceedings, 2006.

[19] M. Samadi, T. Kollar, and M. Veloso. Using the web to interactively learn to find objects. In *AAAI*. AAAI Press, 2012.

[20] T. Sato. A statistical learning method for logic programs with distribution semantics. In *ICLP*, pages 715–729, 1995.

[21] M. Schuster, D. Jain, M. Tenorth, and M. Beetz. Learning organizational principles in human environments. In *ICRA*. IEEE, 2012.

[22] K. Shubina and J. Tsotsos. Visual search for an object in a 3D environment using a mobile robot. *CVIU*, 114(5):535–547, 2010.

[23] K. Sjö, D. López, C. Paul, P. Jensfelt, and D. Kragic. Object search and localization for an indoor mobile robot. *CIT*, 17:67–80, 2009.

[24] M. Tenorth and M. Beetz. KNOWROB - knowledge processing for autonomous personal robots. In *IROS*, pages 4261–4266. IEEE, 2009.

[25] L. Wong, L. Kaelbling, and T. Lozano-Peréz. Manipulation-based active search for occluded objects. In *ICRA*. IEEE, 2013.

[26] Y. Ye and J. Tsotsos. Sensor planning in 3D object search: Its formulation and complexity. *Annals of Mathematics and AI*, 1996.